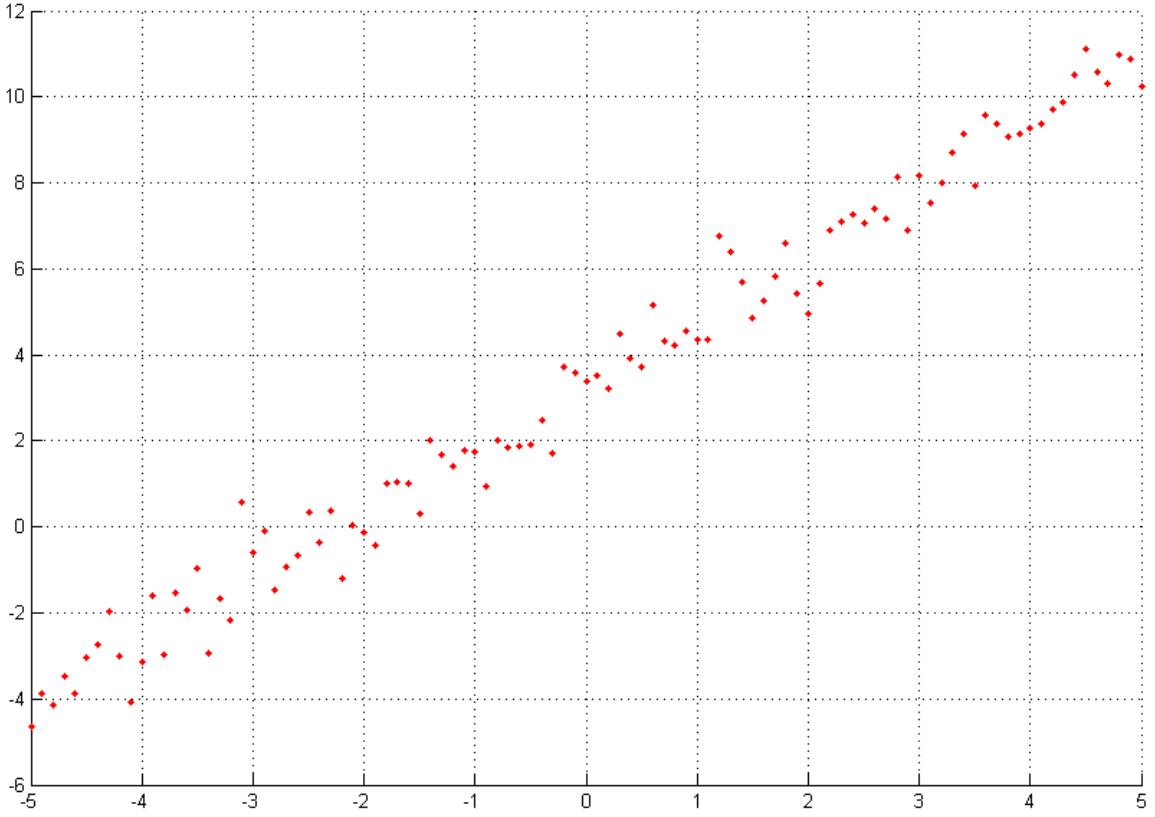
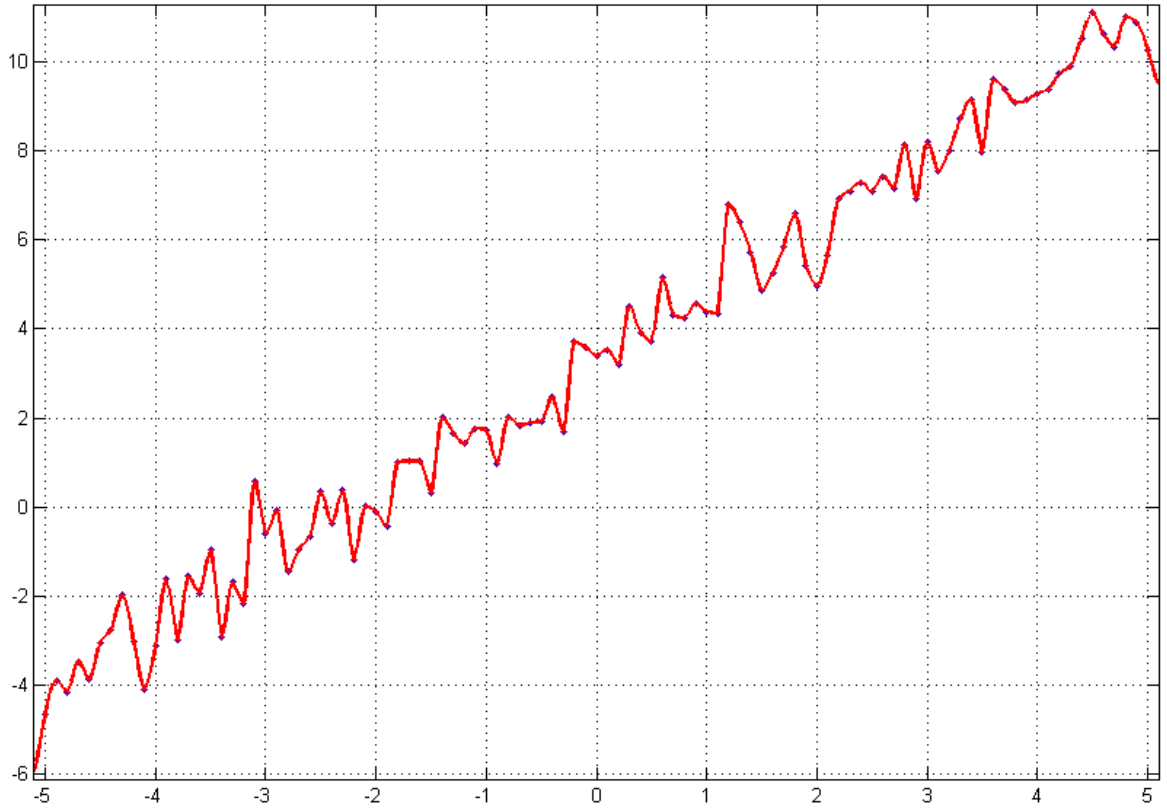
Элементы технического зрения ПРТС

БИБЛИОТЕКА EIGEN. ИТЕРАЦИОННЫЕ МЕТОДЫ. МЕТОД НАИМЕНЬШИХ КВАДРАТОВ.

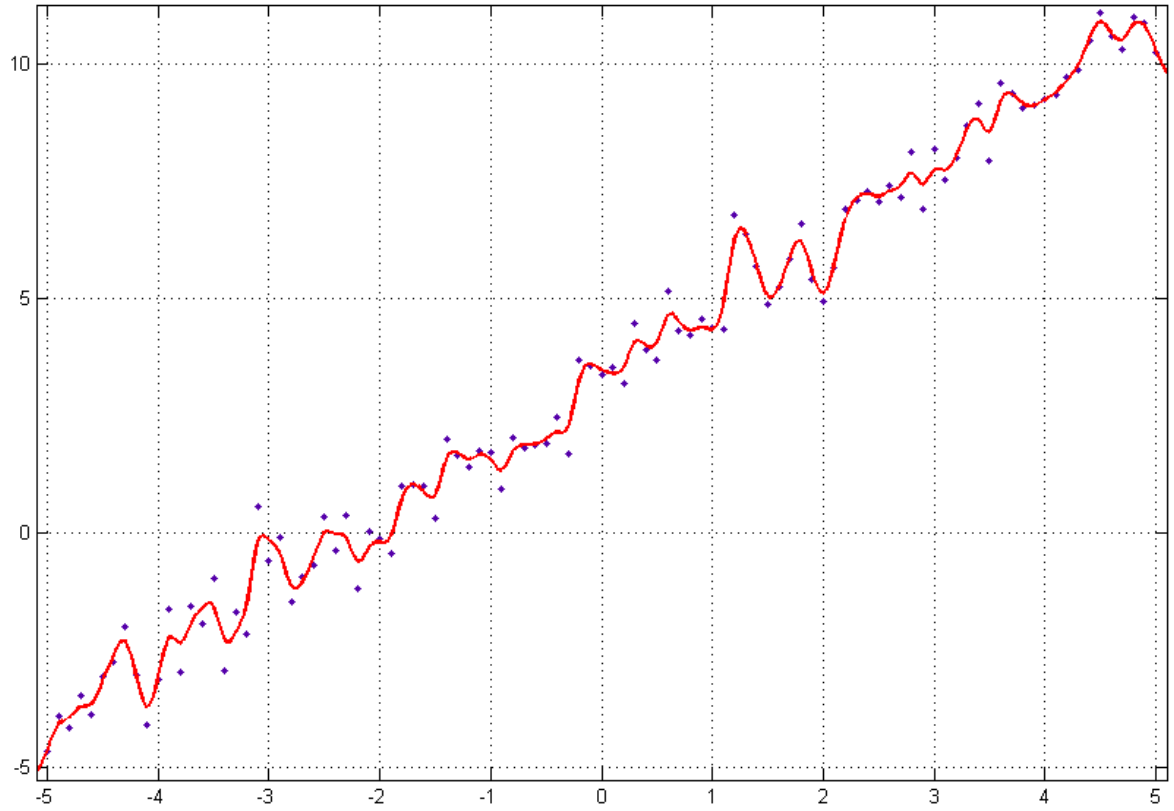
Обработка «зашумлённых данных»



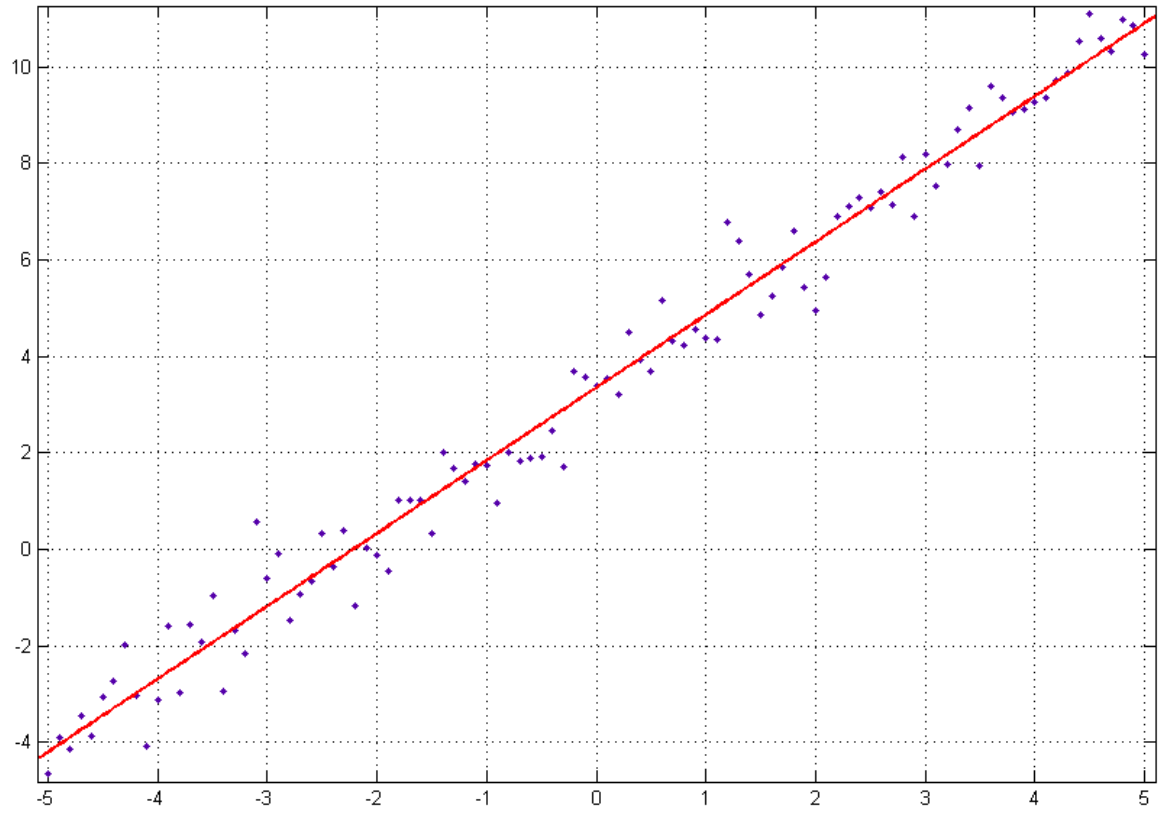
Обработка «зашумлённых данных»



Обработка «зашумлённых данных»



Обработка «зашумлённых данных»



Обработка «зашумлённых данных»

Как выбрать правильное приближение?

Как оценить его параметры?

Как понять, что получилось?

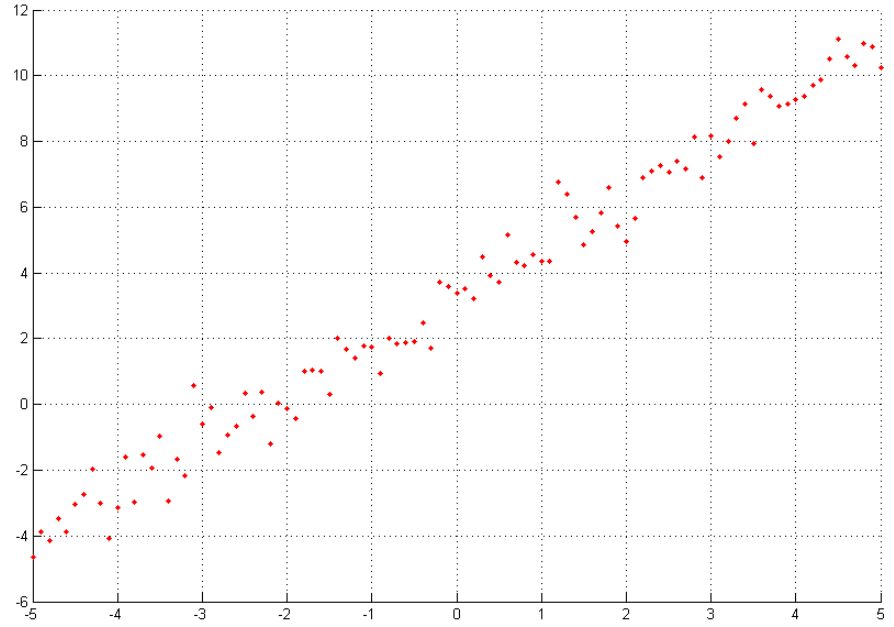
МНК-оценка

Используется для линейных по параметрам моделей

Подразумевает, что модель мы каким-то образом уже выбрали

Например:

- x – данные
- y - значения



МНК-оценка

Модели могут быть разными:

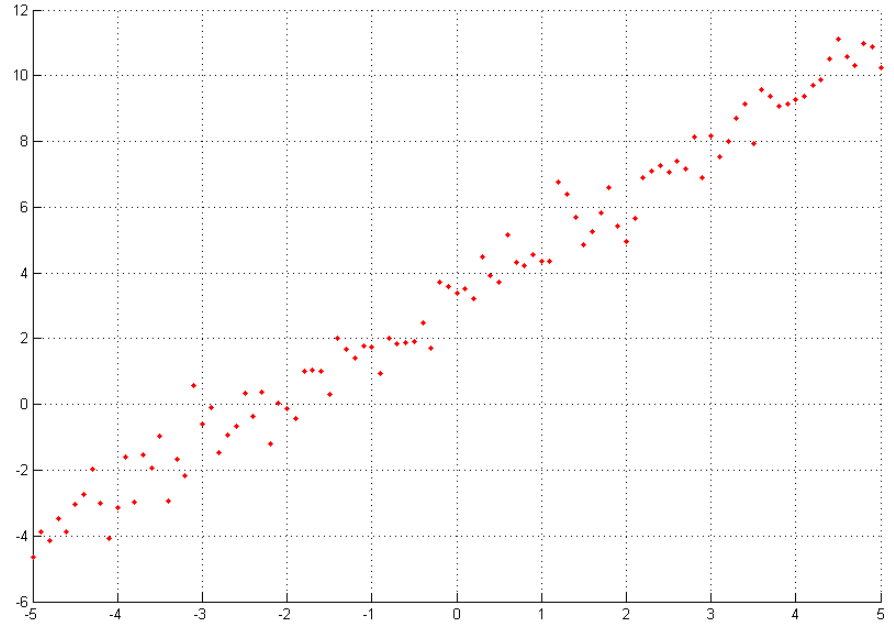
$$u = C = const$$

$$u = A \cdot x + B$$

$$u = A \cdot x^2 + B \cdot x + C$$

$$u = A \cdot e^{Bx}$$

$$u = C_1 \cos(\omega_1 x) + C_1 \sin(\omega_1 x)$$



МНК-оценка

Выберем линейный полином:

$$u = A \cdot x + B$$

- Получим набор значений $u_i = A \cdot x_i + B$
- При этом у нас есть y_i - практически полученные результаты
- Очевидно, что в идеальном случае $u_i = y_i$
- В реальности $r_i = u_i - y_i$ – невязка
- Нужно выбрать A и B так, чтобы невязка была минимальной

МНК-оценка

Будем минимизировать квадратичный функционал:

$$\sum_i r_i^2 \to min$$

Можно записать через вектор:

$$r^T r \to min$$

Условие минимума – равенство нулю первой вариации:

$$\delta r^T r = 0$$

МНК-оценка

Запишем уравнения для компонент r_i:

$$r_i = x_i \cdot A + 1 \cdot B - y_i$$

Можно записать матричное уравнение:

$$\begin{pmatrix} \cdots \\ r_i \\ \cdots \end{pmatrix} = \begin{pmatrix} \cdots & \cdots & \cdots \\ x_i & 1 & y_i \\ \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} A \\ B \\ -1 \end{pmatrix}$$

Тогда функционал можно записать как

$$(A \quad B \quad -1) \begin{pmatrix} \cdots & x_i & \cdots \\ \cdots & 1 & \cdots \\ \cdots & y_i & \cdots \end{pmatrix} \begin{pmatrix} \cdots & \cdots & \cdots \\ x_i & 1 & y_i \\ \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} A \\ B \\ -1 \end{pmatrix} \to min$$

После этого первую вариацию можно записать как

$$\begin{pmatrix} \cdots & x_i & \cdots \\ \cdots & 1 & \cdots \\ \cdots & y_i & \cdots \end{pmatrix}\begin{pmatrix} \cdots & \cdots & \cdots \\ x_i & 1 & y_i \\ \cdots & \cdots & \cdots \end{pmatrix}\begin{pmatrix} A \\ B \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

То же самое

$$\mathrm{F} \cdot \begin{pmatrix} A \\ B \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

где $\mathrm{F} = \begin{pmatrix} \cdots & x_i & \cdots \\ \cdots & 1 & \cdots \\ \cdots & y_i & \cdots \end{pmatrix}\begin{pmatrix} \cdots & \cdots & \cdots \\ x_i & 1 & y_i \\ \cdots & \cdots & \cdots \end{pmatrix}$-матрица 3x3

МНК-оценка

У нас 3 уравнения, но всего 2 переменных

Можно преобразовать у уравнению 2x2:

$$\begin{pmatrix} \cdots & x_i & \cdots \\ \cdots & 1 & \cdots \end{pmatrix} \begin{pmatrix} \cdots & \cdots \\ x_i & 1 \\ \cdots & \cdots \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} \cdots & x_i & \cdots \\ \cdots & 1 & \cdots \end{pmatrix} \begin{pmatrix} \cdots \\ y_i \\ \cdots \end{pmatrix}$$

То же самое:

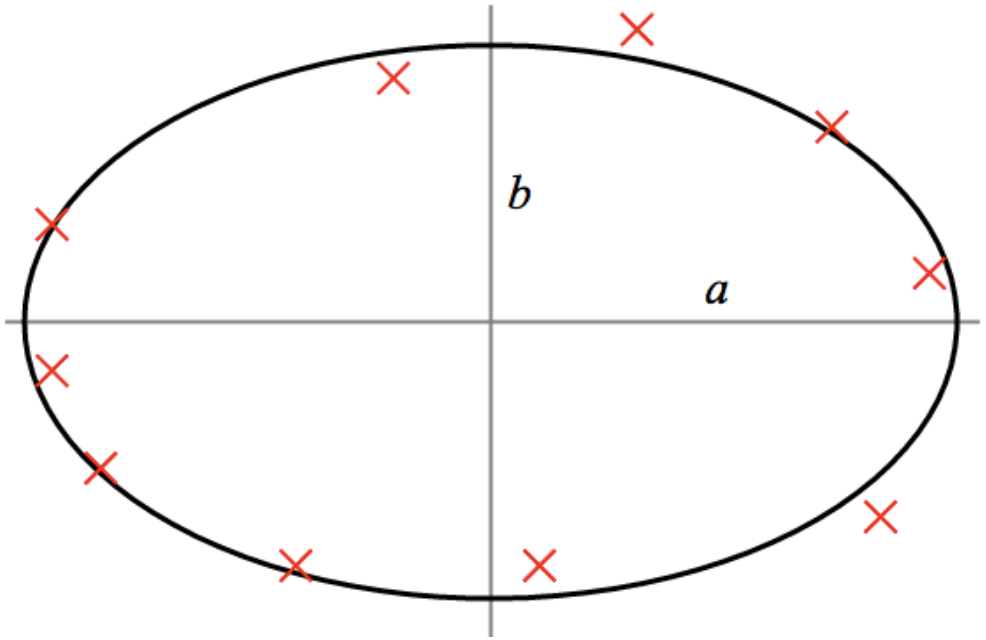
$$G^T G \begin{pmatrix} A \\ B \end{pmatrix} = G^T y$$

Линейность модели

- Уравнение эллипса в общем виде

$$Ax^2 + By^2 + Cxy + Dx + Ey + 1 = 0$$

- Кажется, что уравнение квадратное, но по параметрам оно линейное
- Такое уравнение справедливо для каждой(x_i, y_i)



МНК-оценка

- А если модель экспоненциальная?

$$u = A \cdot e^{Bx}$$

- Можно получить линейную логарифмированием:

$$\ln u = \ln A + Bx$$

- Существует обобщение метода на нелинейный случай (NLLS)
- Тогда строится итерационный процесс с начальным приближением

$$\delta r^T r = 0$$

Применение в Eigen

```
// Создаём матрицу и вектор со случайными значениями
Eigen::MatrixXf A = Eigen::MatrixXf::Random(3, 2);
Eigen::VectorXf b = Eigen::VectorXf::Random(3);

// Применение "в лоб"
cout << "The solution using normal equations is:\n"
<< (A.transpose() * A).lu().solve(A.transpose() * b) << endl;
// Более правильное (через QR-разложение)
cout << "The solution using the QR decomposition is:\n"
<< A.colPivHouseholderQr().solve(b) << endl;
```

МНК-оценка

Как оценить качество полученной оценки?

Самый простой вариант – посчитать норму r_i

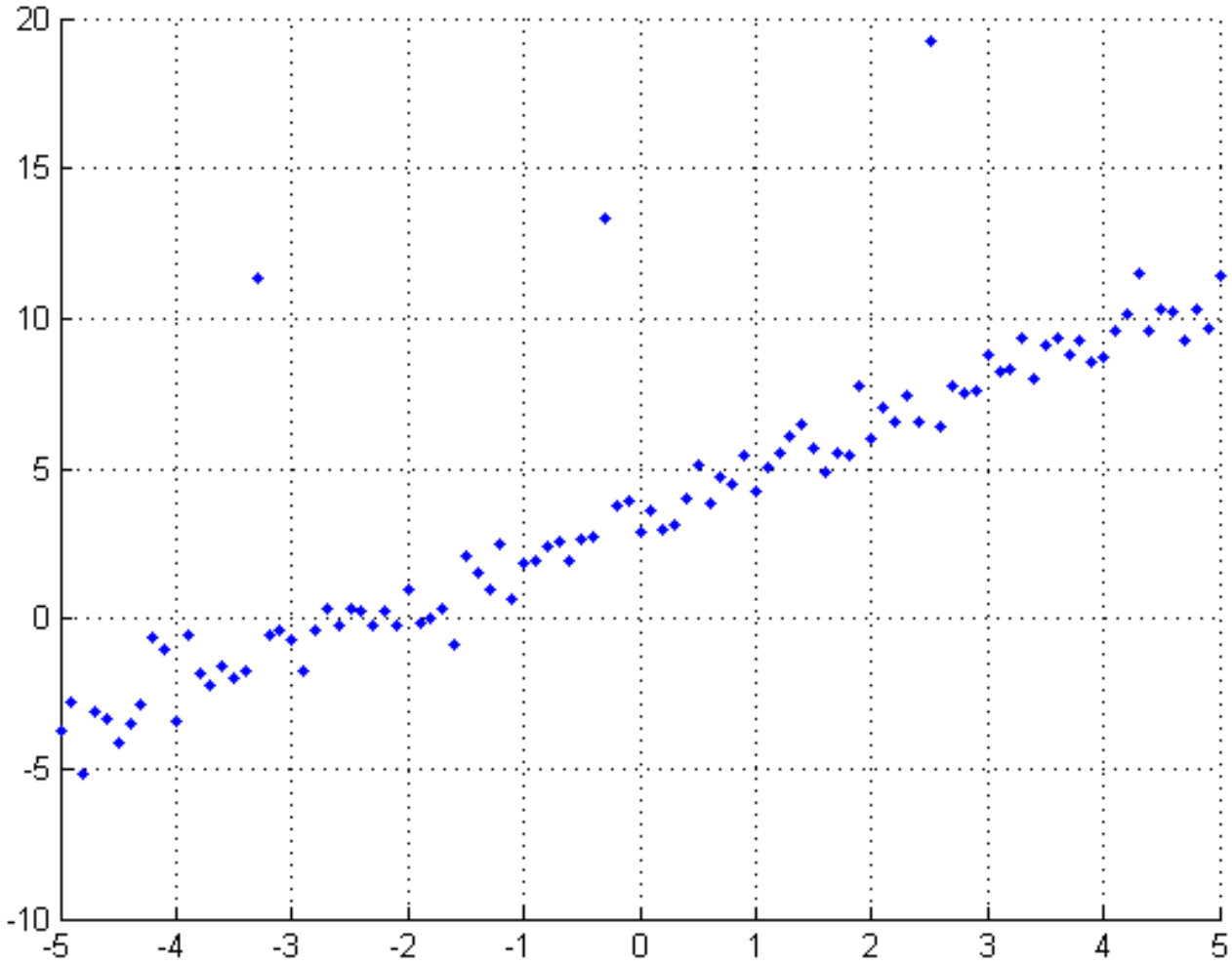
Однако её величина зависит от размерности вектора (числа *n*)

Выход – использовать отношение нормы невязки к норме правой части:
$\delta r = \frac{\|r\|_p}{\|b\|_p}$

Ещё одна проблема – оценка данных с выбросами (outliers)

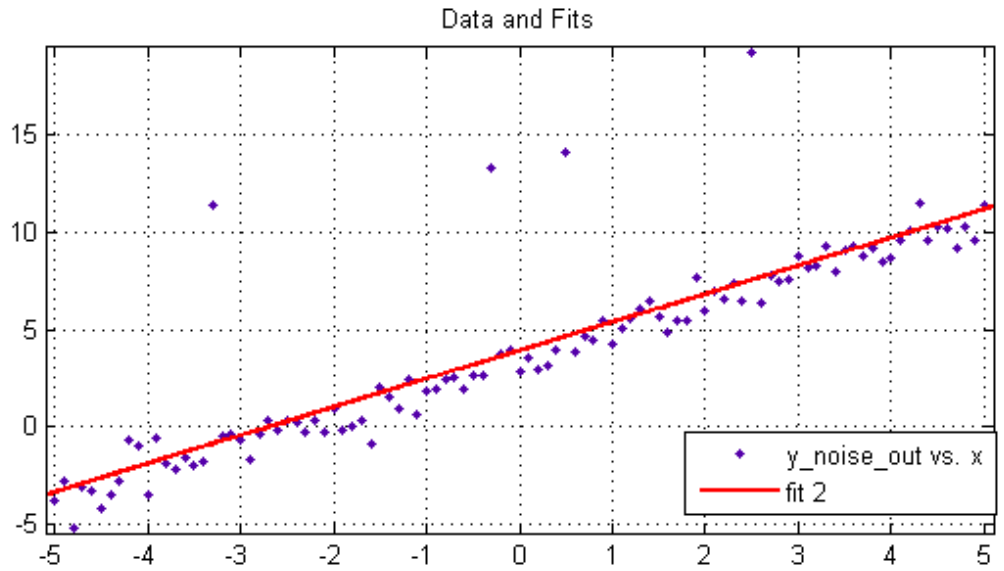
Данные с выбросами

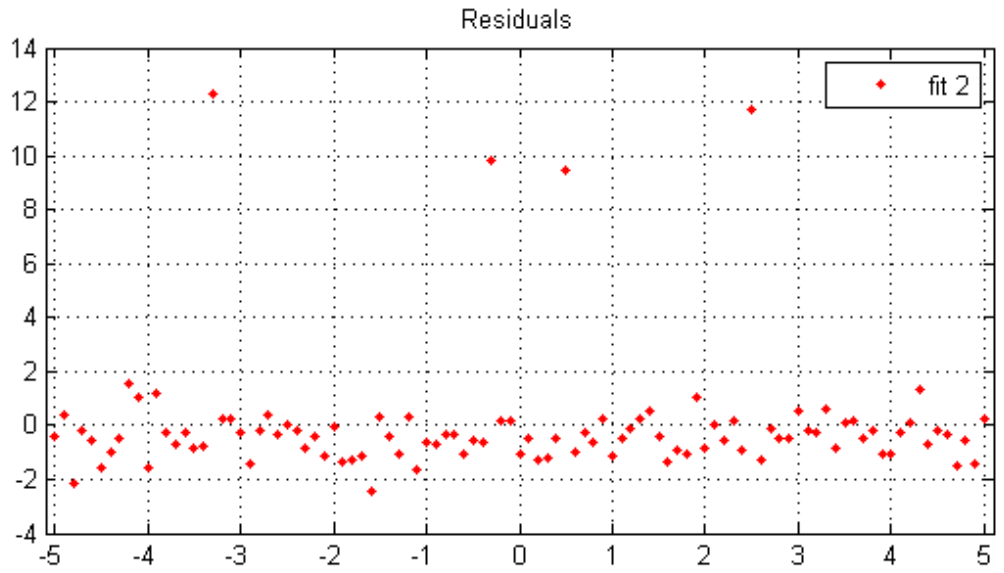
Та же самая линейная зависимость, но теперь в ней есть несколько «неправильных» точек



Данные с выбросами

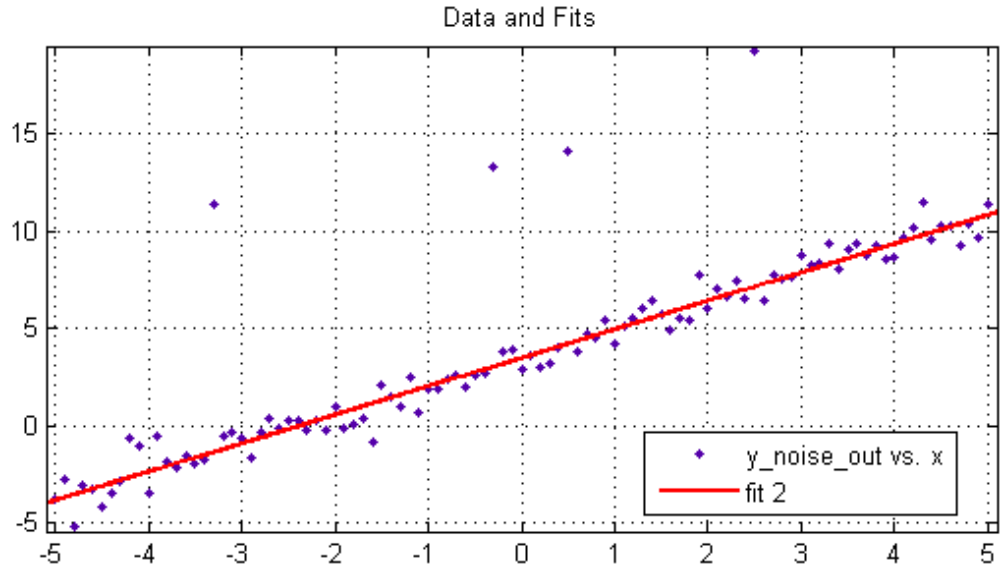
Без учёта выбросов

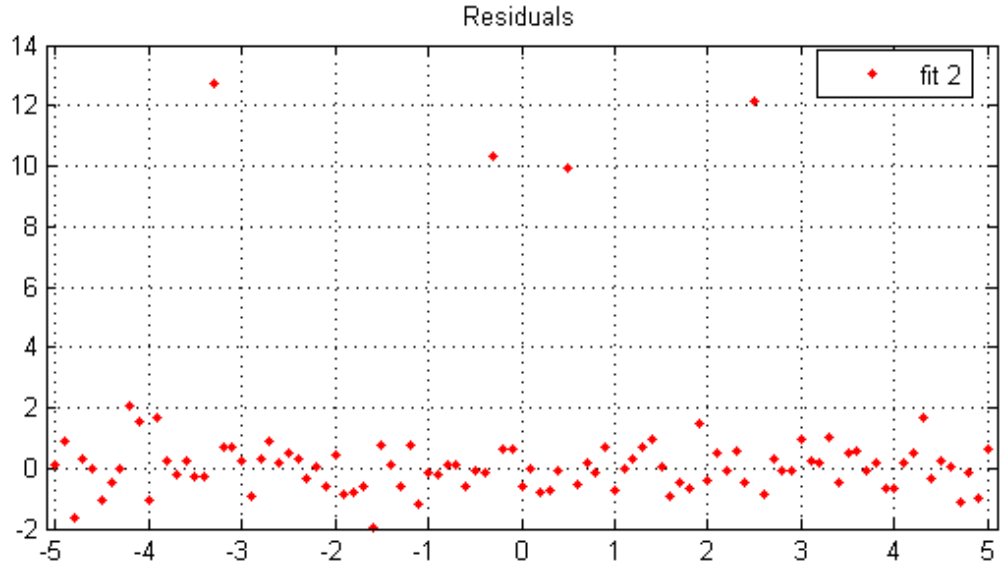




Данные с выбросами

С учётом выбросов





Данные с выбросами

Необходимо использовать робастные методики оценки для исключения выбросов

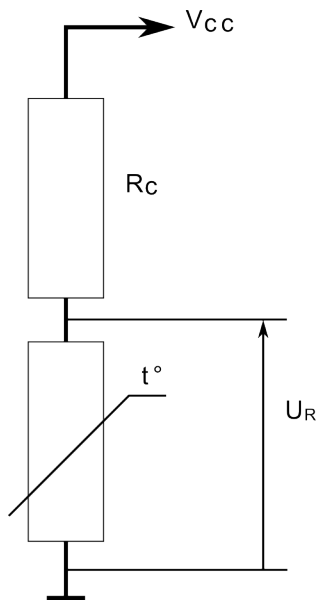
Один из вариантов - RANSAC

Идея в том, чтобы случайным образом выбирать элементы из выборки для построения оценки

RANSAC

- Например: мы оцениваем 2 параметра по 100 отсчётам
- Выберем 100 пар из двух случайных элементов (x_i, y_i)
- Для каждой пары построим δr_j - оценку для j-гипотезы
- Для лучшей пары выберем точки с отклонением больше предельного δ
- Для оставшихся точек построим МНК-оценку
- Вообще говоря, оценку j-гипотезы надо строить тоже по МНК, выбирай заданное число отсчётов

Калибровка AVR для отображения температуры терморезистора



Показания индикатора

$$Z = 1024\frac{U_R}{U_{\text{оп}}} = 1024 \cdot \frac{R}{R_c + R}$$

$$\approx 1024 \cdot \frac{R}{R_c + R_0}$$

Зависимость R - экспоненциальная

$$R = R_0 \cdot 10^{K(T-T_0)}$$

Калибровка AVR для отображения температуры терморезистора

№	Z	T,°C
1	27	71
2	31	64
3	43	52
4	58	41
5	69	33
6	86	23
7	102	17
8	111	12
9	122	2
10	137	0
11	18	87
12	176	-5